TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Jan Durejko

IoT seadmete sõnumite dekodeerimise lihtsustamine kasutajaliidesega

Bakalaureusetöö (9 EAP)

Juhendaja: **Pelle Jakovits**

Tartu 2019

IoT seadmete sõnumite dekodeerimise lihtsustamine kasutajaliidesega

Lühikokkuvõtte:

Bakalaureusetöö eesmärk on kasutajaliidese loomine, kus oleks võimalik sisestada IoT seadme sõnum ja parameetreid valides näha ka tulemust loetava sõnumina. See lihtsustaks seadeldiste lisamist süsteemidesse, kus oleks vaja kuvada seadmetest tulevat infot. See oleks abiks iga päev arenevas IoT valdkonnas, kuna kiirendaks protsessi, mille käigus lisatakse uusi seadmeid ja see omakorda jätaks aega tegeleda uue funktsionaalsuse lisamisega.

Võtmesõnad:

IoT, ReactJS, NodeJS, dekodeerimine

CERCS: P170, Arvutiteadus

IoT sensors messages decoding simplification using user interface

Abstract:

Goal of this thesis is to create user interface what can be used to decode IoT sensor messages. Application should be universal, changing parameters to fit the need and to be able to save the configuration for later use. This makes adding new sensors to systems easier and faster allowing people with no coding experience to do it as well.

Keywords:

IoT, ReactJS, NodeJs, decoding

CERCS: P170, computer science

Sisukord

1. Sissejuhatus

Asjade internet (inglise keeles I*nternet of Things* või lühidalt IoT ja edaspidi ka IoT) on võrgustik, mis koosneb elektroonilistest seadmetest, mis suhtlevad üksteisega või saadavad interneti kaudu välja signaale, nt autod, kodumasinad jpt käivad selle kategooria alla. See on tehnoloogia nähtus, mis on iga päevaga arenev ja rohkematesse seadmetesse jõudev. Ennustatakse, et aastaks 2020 võib neid seadmeid olla juba 50 miljardit [1].

IoT sensorite jaoks on energia efektiivsus ülioluline, kuna tihti kasutatakse neid pikemaajaliselt väliolukordades [2]. Selle tõttu hoiavad nad akut kokku oma tegevuste arvelt. Üheks selliseks tegevuseks on kodeeritud sõnumi saatmine, mida kokku pakitum on sõnum, seda pikem on seadme aku eluiga [3]. Kodeerimata sõnum on jada numbritest ja tähtedest ühe baidi pikkuste juppidena, mis on muus kodeeringus, näiteks kuueteistkümnendsüsteemis. Saadetud sõnum on lühem, aga ka loetamatum. Iga sensoriga tuleb kaasa kasutusjuhend, kus on välja toodud sõnumi seletused, kuidas neid tuleks lahti teisendada ja mis väljadele vastavad antud numbrid nt temperatuur, kiirus jms. Vahel harva on ka kaasas näited. Rakenduse loomisel, mis kasutavad otse IoT sensoritelt tulnud andmeid, tuleb iga uue sensoritüübi jaoks eraldi seadistada andmete dekodeerimine. Sõnumite dekodeerimine võtab aega ja on raske kontrollida mugavalt, kas saabuv info on õige. Hetkel ei ole saadaval universaalset dekoodrit, mis tegeleks ainult dekodeerimisega. Leidub ettevõtteid, kes pakuvad terviklikke lahendusi, nagu seadmete lisamine enda süsteemi, sõnumite dekodeerimise funktsioon ning andmete salvestamine, mille tõttu on neid keeruline kasutusele võtta enda rakenduses, kuna teenusepakkujad soovivad, et kogu tegevus käiks läbi nende süsteemi. Ülevaade nende puudustest esitatakse peatükis 2.2.

Eesmärgiks on lihtsustada sensori seadmete dekodeerimise seadistamist ning automatiseerimist ning luua selle jaoks veebirakendus. Selle rakenduse abil saab sensorist tuleva sõnumi lahti dekodeerida ning salvestada konfiguratsioon hilisemaks käivitamiseks. Esimene kord on vaja sisestada sõnum veebirakendusse, et saaks koostada struktuuri antud sensori sõnumi kohta. Kasutaja saab valida, mis baidi peal on talle oluline info, saab valida ühiku ja määrata väljale nime. Rakendus tõlgib sõnumit interaktiivselt, mis võimaldab vaadata, kas jäi mõnda välja viga sisse või dekodeeritud sõnum vastab sellele, mis on dokumentatsioonis näidatud. Kui dokumentatsiooni näide ühtib rakendusega, on võimalik salvestada konfiguratsioon ehk kui tulevikus sama sensor saadab uue sõnumi, saab kasutada olemasolevat konfiguratsiooni, kuna sõnumi vorming ei muutu, muutub ainult sõnumi sisu. Lisaks on ka eraldi lahter, kus kasutaja ise saab kirjutada koodi, et teha keerulisemat dekoreerimist, mida lihtsa kasutajaliidesega teha ei saa, ning ka seda saab salvestada, et hiljem

uuesti kasutada. Selle lõputöö motivatsioon tuleb Eesti firma Catapult Labs[1] vajadustest, kus autor töötab ning mis tegeleb süsteemiarendusega infrastruktuuride jaoks, mille alla käib ka IoT. Hetkel kirjutavad ettevõtte arendajad sensorite dekodeerimise funktsioone käsitsi ning tulemuste kontrollimine on aeganõudev tegevus, kuna on vaja tervet süsteemi jooksutada testide jaoks. Sellest tuli ka praktilise töö vajadus. Materjalid nagu sensorite sõnumid ja nende testid on tulnud firmalt Catapult Labs, kellel on huvi ka tulevikus selline rakendus kasutusse võtta.

Töö teoreetilises osas annab autor ülevaate sellest, mis on IoT, millised on IoT-s kasutatavad tehnoloogiad ja erinevatest viisidest, mida sensorid kasutavad dekodeeritud andmete edastamiseks. Otsitakse ühiseid jooni erinevate sensorite dokumentidest, mis annaks parema arusaama, kuidas teha universaalsemat lahendust. See töö keskendub aga juhtmevabadele pikamaa seadmetele, mis kasutavad LPWAN (inglise keeles *low-power wide-area networking*) ehk vähese energiatarbimisega traadivaba sidevõrk. Ning räägitakse lähemalt kahest suurimast LPWAN standardist, milleks on LoRa ja Sigfox, kuna populaarsemad sensorid kasutavad neid standardeid ning Catapult Labsi kõik süsteemis olevad sensorid töötavad nende kahe standardi alusel.

Peatükis 1 tutvustatakse lähemalt, mis on IoT ja millised nende kasutusalad on. Samuti tutvustatakse kahte kõige populaarsemat IoT standardit ning nende eeliseid ja puuduseid. Peatükis 2 kirjeldatakse töö praktilisemat poolt, kuidas rakendus toimib ja mis võimalusi see pakub. Kasutades ette antud sensorite dokumente, dekodeerime lahti sensorist tulnud sõnumi ja võrdleme testidega. Peatükis 3 on kirjeldatud rakenduse disain ja arhitektuur. Välja on toodud rakenduse funktsionaalsed nõuded ning kasutusel olevad tehnoloogiad ja miks just need valiti. Lõpetuseks on ülevaade, kuidas kasutada rakendust ning mis võimalused rakendusega on. Peatükis 4 toimub rakenduse testimine, kasutatakse erinevaid sensoreid erinevatest firmadest ning kontrollitakse, kas tulemused ühtivad Catapult Labsi testandmetega.

[1] http://catapultlabs.eu/

2. Tausta ülevaade

Selles peatükis räägitakse lähemalt, mis on asjade internet ehk IoT ning millised on selle peamised kasutusalad näiteks meditsiinis või kodumajapidamises. Antakse ka ülevaade kahest kõige levinumast IoT LPWAN standardist LoRa ja Sigfox ning millal eelistada ühele teist. Lõpetuseks tehakse läbi käsitsi sensori dekodeerimine, mis näitab vajadust selle töö raames loodud rakenduse järele.

2.1 Asjade internet

A. McEwan jt [4] on kirjutanud, et IoT idee on selles, et on suur kogumik vähem võimsaid seadmeid nagu spordikell või bussitabloo. Need ei ole arvutid ega telefonid, vaid seadmed, mis kasutavad internetti info saatmiseks või vastu võtmiseks. Bussitabloo ülesanne on näidata bussi numbrit ja selle saabumise aega. See tähendab, et seadmed tegelevad ühe kindla ülesandega ja teevad seda hästi. Nad ei tohiks teha mitut asja korraga, sest see kõrgendab riknemise võimalust. Seade kas kuvab infot, mis tuleb internetist või teiselt seadmelt, või on ise seadeldis, mis infot saadab. Kui bussitablool oleks küljes veel kaamera, mis loeb saabunud bussi numbrit, et tablool infot uuendada, siis mis juhtub, kui kaamera saab mustaks või õues on udu ja kaamera ei tuvasta enam bussi numbrit.

Kuigi IoT on igapäevaselt üha rohkem populaarsust kogumas, on juba praeguseks suur hulk kasutusalasid väga mitmes valdkonnas, näiteks tervishoid, põllumajandus, tark kodu, tark linn ja erinevad infrastruktuuri osad nagu vee- ja küttesüsteemid [5]. Tabel 1 annab ülevaate peamistest IoT kasutusaladest sellistes valdkondades.

Tabel 1. IoT kasutusalad erinevates valdkondades

Valdkond	**Kasutus**
Meditsiin	Patsiendi jälgimine koos teavitusega, kui midagi peaks valesti olema, spordikellad, veresuhkru mõõtmine, ravimite võtmise jälgimine
Põllumajandus	Põldude viljakus, veetase mullas, loomade toitmine
Tark linn	Veelekked, temperatuur, parkimiskohad, õhusaaste, linna tulede põlemisajad, kütmine
Autod	Tagurdamisandurid, mootoririkke tuvastus, isesõitmine, veojõu kontroll

Goldman Sachs uuringu järgi võib IoT kasutamine meditsiini valdkonnas üksi aidata Ameerika Ühendriikidel kokku hoida üle 300 miljardi dollari, kui viia järjest rohkem patsientide jälgimist IoT seadmete peale üle [6]. Artiklis tuuakse välja sensoreid, mis kontrollivad südamerütmi, astmaravimite tarbimist ja glükoositaseme jälgimist. Teavitades patsiente, kui midagi on vaja teha, vähendades olukordi, kus inimene unustab ravimeid võtta või ei saa piisavalt kiirelt reageerida ja lõpetab raskes seisus haiglas. Sensorid vähendaks seda riski, andes õigeaegseid teavitusi inimesele ja vältides haiglasse sattumist.

Selleks, et seadmed saaks olla kasulikud, peavad nad edastama aga infot ja see peab jõudma kohale kiirelt. Selle jaoks on olemas erinevad standardid info edastamiseks. On olemas standardid, mis on mõeldud lühikese maa läbimiseks, nagu RFID (inglise keeles *radio-frequency identification*), WiFi (inglise keeles *wireless fidelity*) või BLE (inglise keeles *bluetooth low energy*) [7]. Spordikellad kasutavad BLE või siis Bluetooth ühendust, tark kodu kasutab aga WiFi ühendust, kui kõik seadmed asuvad lähestikku ja ühenduvad kodu WiFi võrku.

2.1.2 LoRa

LoRa, mis tähendab "*Long Range*" või siis eesti keeles "pikk maa", on patenteeritud traadita andmete saatmise tehnoloogia. Algselt leiutati Prantsusmaal firma poolt nimega Cycleo, kuid aastal 2012 müüdi patent edasi Semtechile.

Laialt on see kasutusel just sensorites, kuna see võimaldab sensoritel saata korjatavat infot kaugele, kulutades selleks ise vähe akut. Tavaliselt suudab LoRa sensor edastada infot 15km, kui tegemist on tiheda linnaalaga, aga kuni 30 km, kui tegemist on maapiirkonnaga. Aku on mõeldud kestma üle 10 aasta, et peale sensori ülesseadmist ei peaks seda enam hooldama minema [2]. LoRal on üks suurem andmepaketi maht ehk võimaldab edastada 243 baiti infot ja andmeedastuskiirusega kuni 300 kilobitti sekundis [8]. Seega sobib LoRa hästi näiteks GPS seadmetele, kuna võimaldab iga paari sekundi järel edastada infot kiiruse ja asukoha kohta.

2.1.3 Sigfox

Sigfoxi tehnoloogia töötati välja aastal 2010 Prantsusmaal. Hetkel opereerib Sigfox 31 riigis ning see arv suureneb pidevalt [8]. Brian Ray on kirjutanud [9], et Sigfoxi idee on luua IoT standard, mis oleks väga väikse läbilaskevõimega ehk tarbib vähem energiat ja oleks odavam seadmeid toota. Sigfox on kulutanud sadu miljoneid eurosid, et välja töötada oma suletud võrk. See tähendab, et kui osta Sigfoxi seade, ei ole vaja üles seada enda kodust võrku vaid saab selle ühendada Sigfoxi võrku ja kõik toimib. Kuna Sigfoxi strateegia on olla odav ehk odavad seadmed ja odav ühendamise kulu

nende võrku, et meelitada ligi palju kasutajaid, siis tähendab see, et nende enda teenistus on küllaltki väike. Selle tõttu on Sigfox olnud ka majanduslikes raskustes.

Sigfox on pikema edastusraadiusega kui LoRa, kuni 50km maapiirkondades, kuid edastussuurused on väiksemad. Sigfox suudab edastada kuni 12 baiti infot ja kuna on väikse läbilaske võimega - 6 sõnumit tunnis mahuga 12 baiti ja kokku 140 sõnumit päevas - sobib hästi info edasi saatmiseks, mis toimub ainult paar korda tunnis, näiteks ilmateade või parkimiskoha andur ja selle tõttu on Sigfoxi raadiomoodulid umbes 50% võrra odavamad kui LoRa omad [8].

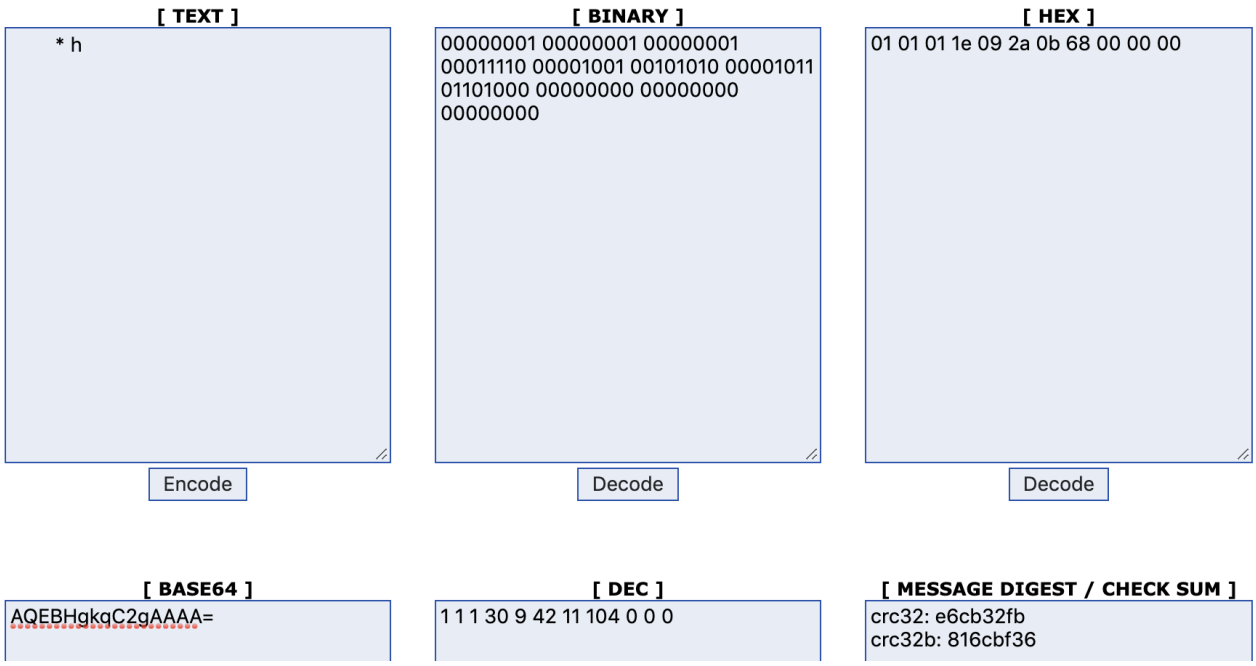
LoRa erinevalt Sigfoxist on standard, mis tähendab, et inimene peab ostma seadmed, millel on sees LoRa raadiomoodul, ning ise ehitama üles võrgu, kus seade saab saata ja vastu võtta sõnumeid. See on plussiks ja miinuseks. Ülesseadmine võtab rohkem aega ja nõuab rohkem kulutusi, kuna on vaja kasutada lüüse (inglise keeles *gateway*) [8]. Samas saab lüüside kasutamist ka plussiks lugeda, kuna on täielik kontroll serveri üle, mille kasutaja peab ise üles seadma. Firmades on turvakaalutustel raske saada nõusolekut, et ühendada oma seadmed kuhugi kolmandasse võrku.

2.1.4 Sensori dekodeerimine

Nii LoRa kui ka Sigfox edastavad sõnumeid kodeeritult, et saaks rohkem infot edastada lühema sõnumiga ja samal ajal energiat kokku hoida, et aku eluiga pikem oleks [3]. Sõnumi dekodeerimiseks on kõigepealt vaja sõnumi formaadist aru saada. Esmalt tehakse läbi näide, kuidas käsitsi dekodeerida sõnumit, mis näitab, kui ajamahuka protsessiga on tegu. Näitena on kasutusel Netvoxi temperatuurisensor. See edastab infot aku, temperatuuri ja niiskuse kohta, mis on näha joonisel 2. Sensor on mõõtnud temperatuuriks 23,46 kraadi ning edastab sõnumi kodeeritud kujul. Kasutades Netvoxi dokumenti ja erinevaid konverterid vaatame, kas vastus tuleb õige.

Alguses tuleb võtta sensorist tulev sõnum ja ta tükeldada vajalikeks osadeks, et saaks edasi minna. Siin kasutatakse näitena Netvox LoRaWan temperatuurisensorit ning sõnum, mille ta edastab, on „ AQEBHgkqC2gAAAA=“. Kui sõnum on tükeldatud, on võimalik kasutada internetis leiduvaid Hex konvertereid, näiteks ScadaCore Online Hex Converter [10], mis lubab sisestada hex ehk kuueteistkümnendsüsteemi kodeeringuga sõne, mis tõlgitakse omakorda 14sse erinevasse vormingusse.

Vahel ei ole sensorist tulev sõnum aga Hex kodeeringus, nagu ka näites, vaid hoopis base64. Sellisel juhul on vaja sõnum enne viia hex kujule. Selle jaoks on internetis sadu lehekülgi, aga käesolevas töös on kasutusel darkByte [11], sest see pakub korraga kõige levinumaid vorme.

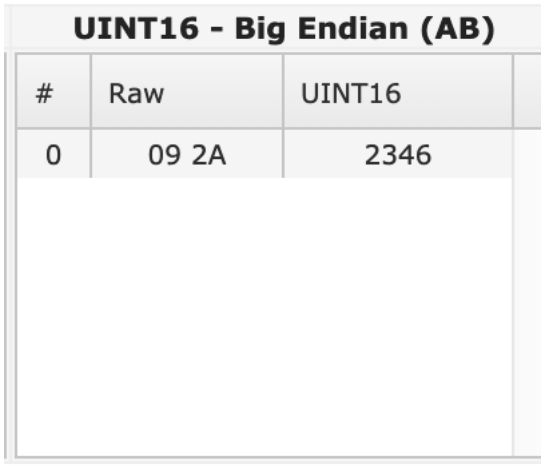


Joonis 1. Ekraanipilt base64 kodeering Hexi.

Esiteks tuleb tõlkida sensori sõnum hex kodeeringusse (näidatud joonisel 1) ja tulemuseks on „01 01 01 1e 09 2a 0b 68 00 00 00“, mis on nüüd sobivas formaadis, et edasi liikuda. Joonisel 3 on näha, et *Device, Device Type, ReportType* tulbad ja *NetvoxPayLoadData* veerus *Battery* on kõik ühe baidi pikkused. *Temperature* ja *Humidity* on aga kahe baidi pikkused. Temperatuuri jaoks on siis „09 2a“ ehk neljas bait ja pikkus on kaks baiti. Kui lisada see ScadaCore hex muundurisse, siis tulemuseks on 2346 (näidatud joonisel 3). Kuna juhendis (näidatud joonisel 2) on kirjas, et *unit* on 0,01, on saadud arv vaja läbi korrutada 0,01ga ehk tulemuseks on 23,46, mis on õige vastus.

Device	Device Type	ReportType	NetvoxPayLoadData			
R711	0x01/ 0x0B/ 0x	0x01	Battery(1Byte)	Temperature (Signed2Bytes, unit: 0.01)	Humidity (2Bytes, unit:0.01)	Reserved (3Bytes, fixed 0x00)

Joonis 2. Ekraanipilt Netvox temperatuurianduri dekodeerimise juhendist [12].



Joonis 3. Ekraanipilt ScadaCore hex muundur.

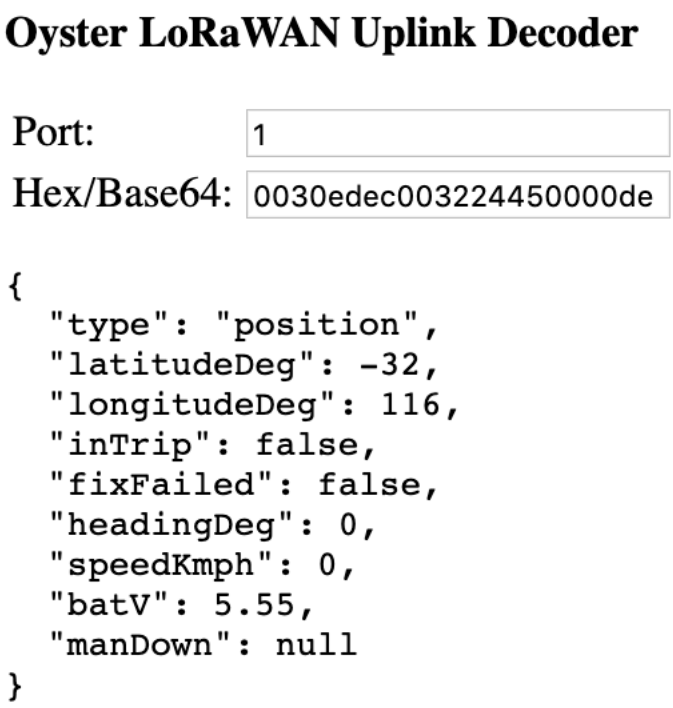
Nüüd tuleks samamoodi läbi teha see niiskuse kui ka akuga ning need vastused salvestada. Kui aga sensorist tulev info maha salvestatakse, et saaks hiljem võrrelda, milline nädal oli kõige soojem, siis käsitsi tegemine läheb väga ajamahukaks. Järgnevates peatükkides tutvustatakse viise, kuidas seda protsessi kergendada, kasutades erinevaid internetis leiduvaid lahendusi.

2.2 Olemasolevad lahendused

Käesolevas alampeatükis tutvustatakse juba olemasolevaid lahendusi, võrreldakse neid käesoleva tööga ja jõutakse järeldusele, kuidas teha praeguseid lahendusi paremaks või võimaldada rohkem paindlikkust.

2.2.1 Digital Matter

Digital Matter[13] on loonud üksikuid veebirakendusi, kus on võimalik sisestada sensori sõnum, mis siis selle lahti dekodeerib (näidatud joonisel 4). Tegemist on firmaga, kes loob GPS sensoreid, mis edastavad infot näiteks asukoha ja kiiruse kohta [14]. Siin olevas näites kasutatakse Oyster LoRaWan Uplink seadme dekoodrit, kuna seadmele vastavad testid on Catapult Labil olemas, mis teeb andmete valideerimise täpsemaks.



Joonis 4: Digital Matteri „*Oyster LoRaWAN Uplink Decoder“*.

Digital Matteri idee on sarnane selle töö käigus tehtava lahendusega. Kasutajal on rakendus, kuhu saab sisestada Hex või base64 vorminguga sõnumi ja rakendus loeb välja vajalikud väljad ning dekodeerib need. Nagu joonisel 4 on näha, ei ole võimalik seal midagi muuta, saab ainult sisestada sõnumi. See piirab selle kasutusvõimalusi ainult seda tüüpi sensori jaoks. Oyster LoRaWAN on GPS jälgija, aga kui aasta pärast sellist sensorit enam osta ei saa ja on saadaval teiste firmade GPS jälgijad, siis on juba vaja uut dekoodrit. Sellele lisaks ei ole võimalik antud lahendust kuidagi ise kasutada, tegemist on veebirakendusega, mis kuvab küll dekodeeritud sõnumi, aga kuna kood ei ole kättesaadav, on vaja sõnumist ise aru saada ning antud rakendust saab kasutada ainult kontrolliks.

Rakendus sobib inimesele, kellel on kodus üks selline sensor ja huvi valmis programmeerida rakendus, mis sõnumitest aru saaks. Kasutades Digital Matteri rakendust, saab ta kontrollida, kas sai sensori dokumendist õigesti aru. Selle töö eesmärk on luua sarnane rakendus, mis aga lubaks universaalsemalt dekodeerida sõnumeid, nii et saab üheskoos kasutada erinevate firmade sensoreid, kuid jättes alles kergesti hoomatav kasutajaliides.

2.2.2 The Things Network

The Things Network[2] on võrgustik, kus on võimalik lisada LoRa sensoreid, neid hallata ja salvestada nendest tulevat informatsiooni. Tegemist on väga mugava keskkonnaga, kui kasutaja on

[2] https://www.thethingsnetwork.org/

nõus ühendama oma sensorid nende avaliku võrguga. Sellisel juhul on ka rakenduse kasutamine tasuta, kuid ei ole võimalik hallata servereid, mis tegelevad andmete hoiustamisega ehk ei saa kunagi kindel olla, et süsteem 100% ajast üleval on või kui turvaliselt salvestatakse andmeid. Pole ka teada, kas sensorite info on kõigile kättesaadav. See on sobilik näiteks koduse ilmajaama jaoks, mis ei ole nii kriitiline, et iga minut uut infot töötlema peaks ning andmelekked ei kujutaks ohtu privaatsusele.

Kui suurem ettevõte soovib lisada suures koguses sensoreid võrgustikku, siis on võimalik osta neilt litsents, mis lubab seada üles enda serveri. See on küll kulukas, aga sellisel juhul on kliendil kontroll oma andmete üle, kui liiklus ei käi enam läbi The Things Networki.

Üldiselt on tegemist mugava keskkonnaga, kus on isegi võimalik kirjutada enda dekooder, mis on kujutatud joonisel 5. Miinuseks on see, et pakutakse tervet platvormi. Ei saa kasutada valitud funktsionaalsust, mida integreerida lokaalse süsteemiga. Peab lisama oma seadme nende võrku ehk sensoritest tulenev info jääb nende serveritesse. Lisaks on ka piirang, mitu seadet lisada saab. Seda kõike saab küll vältida, aga siis on tegemist juba tasulise teenusega. See on probleemiks, kui on enda süsteem juba loodud, aga sooviks lihtsalt võimalust dekodeerida sensoreid. On võimalik need sensorid lisada rakendusse ja sissetulev info edasi saata, aga see tähendab topelt tööd ja ka seda, et kasutaja sensori andmed on nende avalikus serveris.

Joonis 5: The Things Network’i dekooder.

2.2.3 Olemasolevate lahenduste puudujäägid

Sarnaseid rakendusi leidub, aga pole ühtegi täpselt sellist, mille järgi on Catapult Labil vajadus. Olemasolevad lahendused on kas liiga mahukad koos paljude lisadega, mida alati vaja ei ole, või on mõeldud dekodeerima ühe kindlat tüüpi sensori sõnumeid.

Selle töö raames tehakse võimalikult universaalne lahendus, mis oleks pistikmoodul, mida saab kasutada sensori sõnumi dekodeerimiseks. See võimaldaks kasutajal salvestada infot enda serveris, mis hoiab andmeid privaatsena. Samas oleks rakendus piisavalt paindlik, et saab kasutada erinevat liiki sensoreid ning nende dekodeerimine oleks kõigil sama mugav ja kerge, andes täieliku kontrolli kasutajale, kuidas ta soovib neid andmeid ja mis formaadis.

3. Rakenduse disain ja arhitektuur

Selles peatükis kirjeldatakse loodud rakenduse funktsionaalsust. Esmalt antakse ülevaade rakenduse nõuetest ja arhitektuurist, seejärel kirjeldatakse, kuidas sensori sõnum lisada rakendusse ja kuidas dekodeerida seda kasutajaliidesega. Järgmiseks kirjeldatakse, kuidas koostada funktsioon, mis dekodeeriks sõnumi ja näidatakse lisavõimalusi, mis tulevad kaasa funktsiooni koostamisega. Viimaseks kirjeldatakse, kuidas salvestada rakendusest tulevat konfiguratsiooni ja kuidas see hiljem käivitada arvuti käsurealt.

Koodi käivitamise eelduseks on, et arvutisse on paigaldatud Node.js. Antud töös kasutati Node.js versiooni 10.15.1[3].

3.1 Nõuded

Funktsionaalsed nõuded näitavad tegevusi, mida kasutaja peaks saama teha loodud veebirakendusega. Tabelis 2 on välja toodud kõik funktsionaalsed nõuded ning tabelis 3 mittefunktsionaalsed nõuded.

Tabel 2. Funktsionaalsed nõuded

ID	Funktsionaalne nõue
1	Kasutaja saab ette anda sensorist tuleva sõnumi.
	Veebirakenduses on lahter, kuhu kasutaja saab kirjutada sensori sõnumi.
2	Kasutaja saab määrata ploki pikkuse baitides.
	Rakendus peaks looma igale sõnumi baidile vastava rea ning kasutaja saab valida mitmest baidist ta infot loeb, kas ühest, kahest või kolmest järgnevast.
3	Kasutaja saab määrata kordaja, millega sõnum läbi korrutatakse.
	Kasutaja sisestab arvu, millega dekodeeritud sõnum läbi korrutatakse, et saada tulemus.
4	Kasutaja saab dekodeerimisel määrata tulemuse ühiku.
	Kasutaja saab lisada ühiku, mis lisatakse dekodeeritud tulemusele juurde.
5	Kasutaja saab määrata nime igale baidilt tulevale sõnumile eraldi.
	Kasutaja saab sisestada nime, mis ilmub väljundisse väärtuse nimena ja tulemus selle järgi JSON formaadis.
6	Kasutaja näeb dekodeeritud sõnumit kui väljale on nimi lisatud.
	Kui kasutaja on lisanud väärtusele nime, siis kuvatakse tulemus, et ei oleks võimalik nimeta väärtusi lisada.

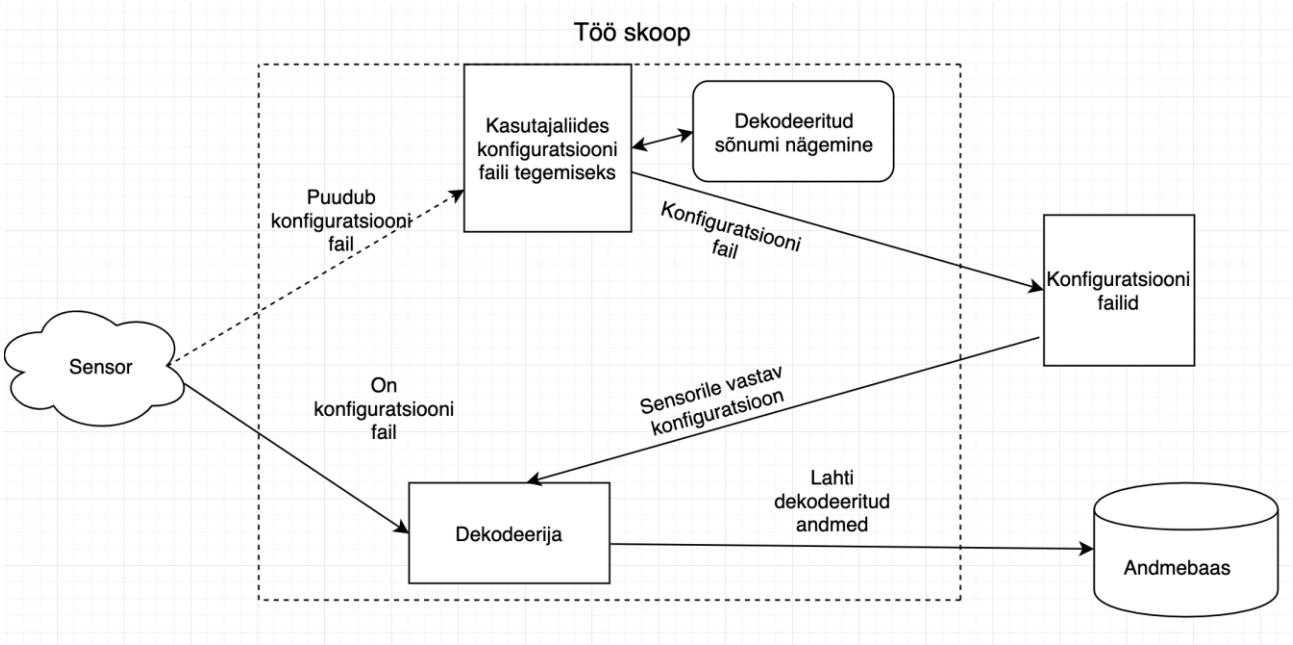
[3] https://nodejs.org/dist/v10.15.1/

ID	Funktsionaalne nõue
7	Kasutaja saab kirjutada ise JavaScript funktsiooni, mis sensori sõnumi dekodeerib
	Kasutajal peaks olema võimalik ise kirjutada sõnumi dekodeerimise funktsioon ning seda ka jooksutada.
8	Kasutaja saab salvestada dekodeerimise konfiguratsiooni .JSON või .js failina
	Kui kasutaja kirjutab ise funktsiooni, siis salvestatakse see .js failina, aga kui kasutatakse kasutajaliidest, siis on konfiguratsiooni fail .JSON failis.
9	Kasutaja saab konfiguratsiooni faili käivitada käsurealt uue sõnumiga.
	Kasutaja saab käsurealt lisada konfiguratsiooni faili (nii .js kui .JSON) ja sensori sõnumi ning see dekodeeritakse vastavalt konfiguratsioonile.

Tabel 3. Mittefunktsionaalsed Nõuded

ID	Mittefunktsionaalne Nõue
1	Rakendus peab olema graafilise kasutajaliidesega.
2	Rakendus peab olema kergesti ligipääsetav ehk veebirakenduse kujul.
3	Rakendus peaks töötama Google Chrome, Mozila Firefoxi ja Safari brauseritega.
4	Rakendus ei tohi kauem laadida kui 3 sekundit.

3.2 Arhitektuur



Joonis 6. Diagramm arhitektuurist ja selle osa kogu lahenduses.

Joonisel 6 on kujutatud selle töö käigus välja pakutud arhitektuur. Kui sensorile vastav konfiguratsioonifail puudub, siis koos dokumentatsiooni ja ühe sõnumiga tuleb see kasutajaliidest kasutades luua. Kasutajaliidest kasutades on võimalik dünaamiliselt näha lahti dekodeeritud sõnumit, et kontrollida, kas tulemus on õige. Kui sensori sõnum on dekodeeritud, saab salvestada sensori konfiguratsiooni. Igal järgneval sõnumil, mis saabub sensorist, saab salvestatud konfiguratsioonifaili abil dekodeerida käsurealt läbi dekoodri, mis võtab sisenditeks konfiguratsioonifaili ning sõnumi ja tagastab dekodeeritud sõnumi, nagu sai seadistatud kasutajaliideses. Hetkel on valmis osad, nagu kasutajaliides konfiguratsioonifaili loomiseks ning dekooder, mis on käivitatav käsurealt. Automaatne info edastamine on plaanis edasiarendusena, kui on vaja rakendus integreerida süsteemi osaks.

3.3 Rakenduse tehnoloogiad

ReactJS

ReactJS on Facebooki poolt loodud JavaScripti raamistik, mis on loodud kasutajaliideste loomiseks [15]. See on mõeldud ka looma ühe lehekülje rakendusi, kuna hoiab infot olekutes ja ei pea lehti uuesti laadima, kui info muutub. Antud töös on kogu kasutajaliides koos funktsionaalsusega loodud kasutades ReactJS. Kasutatakse ka Reacti teeki nimega buffer [16]. Tegemist on Node.js standard funktsionaalsusega, mis lubab teha mugavalt teisendusi erinevate kodeeringute vahel. ReactJS sai valitud, kuna töö autoril on kõige rohkem kokkupuudet just selle raamistikuga. Teiseks, ReactJS ei nõua lehe uuesti laadimist, kui info rakenduses muutub ehk on võimalik luua dünaamiline, kiire ja kasutajasõbralik veebirakendus.

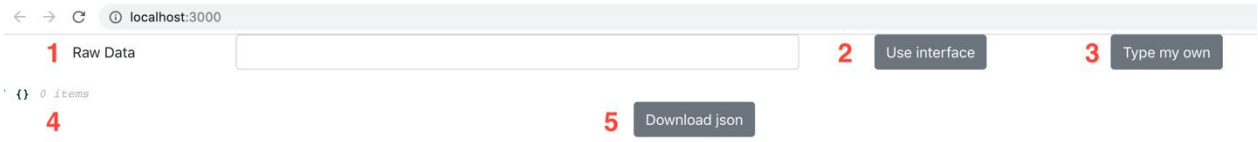
Node.js

Node.js on vabavaraline JavaScripti käitussüsteem, mis toimib platvormist sõltumatult [17]. Node.js võimaldab nii luua serveripoolset rakendust kui ka luua käsurealt käivitavaid skripte.

Selle töö jaoks oli Node.js kõige loogilisem lahendus, kuna veebirakenduses saab ise kirjutada funktsiooni, mis sensorist tulevat sõnumit dekodeerib. Kuna see funktsioon on kirjutatud JavaScriptis, saab selle otse importida Node.js skripti ja jooksutada ilma midagi muutmata. See oli ka peamine põhjus, miks selle töö raames sai Node.js valitud, teistes keeles poleks funktsiooni käivitamine olnud võimalik ilma midagi muutmata.

3.4 Kasutajaliides

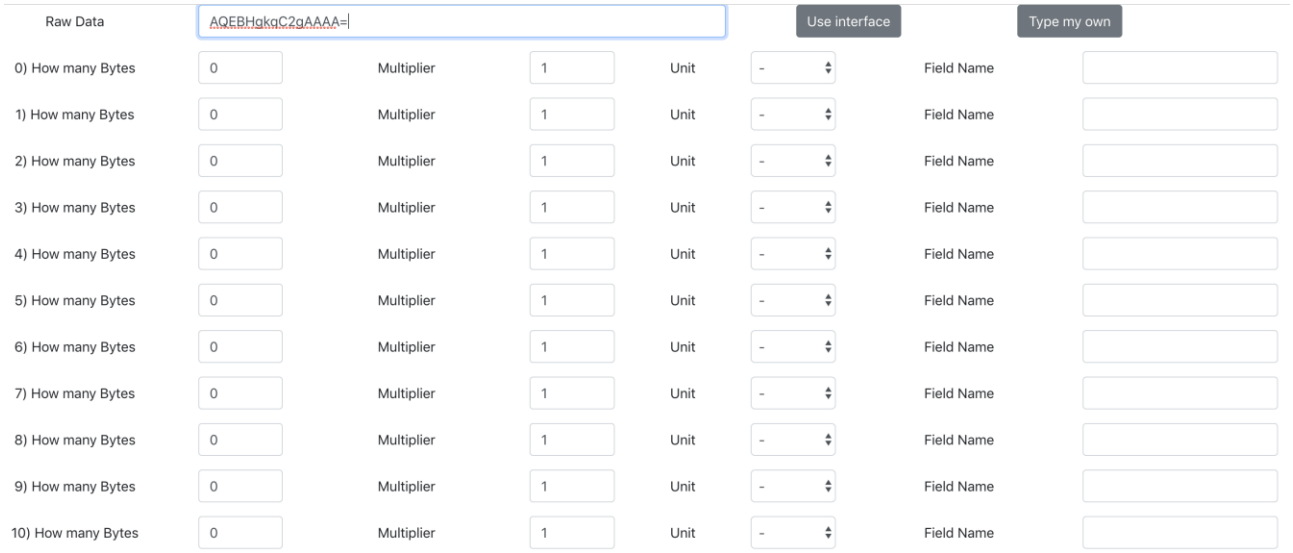
Rakenduse kasutamiseks on see esmalt vaja käivitada. Selleks tuleb minna kausta, kus asub rakendus ja kirjutada käsureale „npm install“, kui on esmane käivitus ja seejärel „npm start“. Rakendus peaks avanema automaatselt brauseris, aga kui seda ei juhtu, tuleb avada brauser ja minna lehele http://localhost:3000/. Ette peaks ilmuma avaekraan nagu joonisel 7 näha.



Joonis 7. Vaade avalehest.

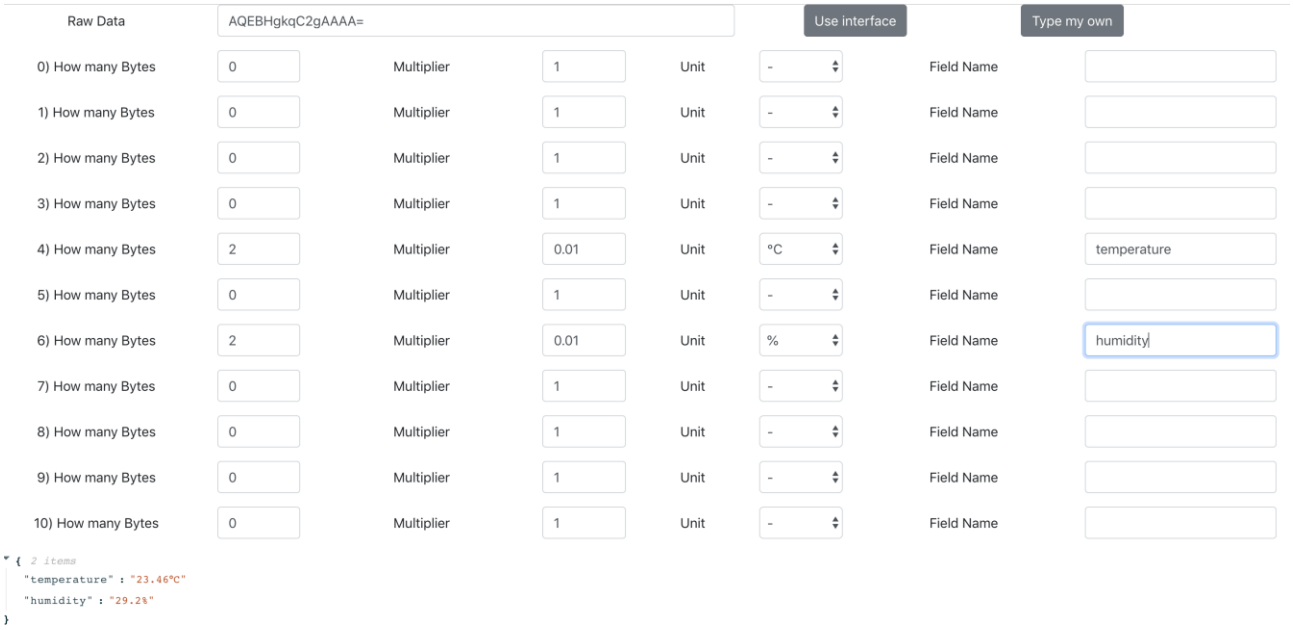
Joonisel 7 on välja toodud viis numbrit, nende kirjeldused on järgnevad. Number 1, tekstiga *Raw Data* on lahter, kuhu on võimalik kirjutada sensorist tulev näitesõnum, mida soovitakse dekodeerida. Number 2 ja 3 on valikud, kas soovitakse kasutada kasutajaliidest (number 2) või soovitakse ise koostada funktsiooni (number 3). Number 4 on ära töödeldud sõnum, mida on näha siis, kui sõnum on dekodeeritud. Dekodeeritud sõnum on JSON formaadis ehk objekt, mis on veebiarenduses ning ka IoT’s kõige levinum andmete hoiustamise viis, kuna on väga erineva struktuuriga andmeid ning ei ole ühtset struktuuri, siis JSON teeb selle arusaadavamaks. Number 5 on nupp, millega on võimalik salvestada konfiguratsioon, et hiljem käivitada käsurealt uue sõnumi peal, mille formaat on sama.

Peatükis 3.1.3 tehti läbi näide kasutades Netvox temperatuurisensorit. Järgnevalt näidatakse rakenduse funktsionaalsust kasutades sama sõnumit. Esmalt lisame *Raw Data* lahtrisse sõnumi „AQEBHgkqC2gAAAA=“. Rakendus seejärel tükeldab sõnumi ära ühe baidi pikkusteks osadeks ja rakendus loob read kasutajaliidesesse, igale reale vastab üks bait nagu joonisel 8 näha.



Joonis 8. Kasutaja lisatud sõnum on tükeldatud 11 osaks.

Sõnum võib olla nii hex kui ka base64 kodeeringus, rakendus oskab selle teha vastavalt kodeeringule vajalikeks osadeks. Järgmisena saab ära täita väljad, kasutades selleks Netvoxi dokumenti, nagu joonisel 2 näidatud. Hetkel leiame temperatuuri ja õhuniiskuse. Kuna aku arvutamine on keerulisem valem, on selle jaoks vaja koostada hiljem funktsioon, seda seletatakse lahti järgmises osas. Dokumendis on näha, et temperatuur on 4. tulp, kuna alustame baitide puhul lugemist nullist. Samuti on temperatuuri pikkus 2 baiti, see on kirjas kui *Signed2Byte*, *multiplier* on 0,1 ja *unit* on °C. Täidame samamoodi ka ära õhuniiskuse. Meelde tuleb jätta, et kuna temperatuuri pikkus on 2 baiti, siis õhuniiskust alustame kuuenda baidi pealt. Kasutajaliides peaks välja nägema nagu joonisel 9.



Joonis 9. Kasutajaliides, kui täita väljad Netvoxi dokumendi järgi.

Kui kirjutada midagi välja *Field Name* ehk nime lahtrisse, ilmuvad alla dekodeeritud lahtrid, nagu ka peatükis 3.1.3 saime, et temperatuur peab olema 23,46 kraadi, mis ka kasutajaliides meile praegu näitab. Õhuniiskuseks saame 29,2%, mis on ka õige tulemus, kui vaadata joonist 10. Kutsutakse välja *decode* sõnumil „AQEBHgkqC2gAAAA=“ ja all on toodud kõik väljad koos väljade väärtustega.

```
expect(sensor.decode( base64EncodedMessageString: 'AQEBHgkqC2gAAAA=',
  {
    deviceType: 'R711',
    metaData: [
      {
        title: 'temperature',
        unit: '°C',
        value: 23.46
      },
      {
        title: 'humidity',
        unit: '%',
        value: 29.2
      },
```

Joonis 10. Catapult Labsi test Netvox sensori sõnumile.

```
Jans-MacBook-Pro:praktilineOsaB jandurejko$ node decoderB.js export.json AQEBHgkqC2gAAAA=
{ name: 'temperature', value: 23.46, unit: '°C' }
{ name: 'humidity', value: 29.2, unit: '%' }
```

Joonis 11. Rakenduse käivitamine käsurealt eelnevalt salvestatud konfiguratsioonifailiga.

Järgnevalt tehakse läbi kasutajaliidesest tuleva konfiguratsiooni salvestamine. Selleks on vaja eelnevalt ära täita kasutajaliideses lahtrid, kontrollida, et tulemused sobivad ja seejärel vajutada nupule *Download Configuration*, mis salvestab kasutaja arvutisse konfiguratsioonifaili, mis tuleks liigutada samasse kausta, kus asub praktilise osa B fail decoderB.js. Seejärel kirjutada käsureale „node decoderB.js export.json {sensori sõnum}” nagu ka joonisel 11 näha. Käsureale ilmub selle peale dekodeeritud sõnum, nagu kuvati kasutajaliideses, aga nüüd on võimalik sõnumit muuta ja ei pea käsitsi enam kasutajaliideses uuesti väljasid täitma.

3.5 Kasutaja defineeritud funktsioon

Järgnevalt tegeletakse ise funktsiooni kirjutamisega. Osad sõnumid sisaldavad valemeid, et saada kätte korrektne vastus. Sellisel juhul on võimalik kirjutada ise funktsioon, kasutades selleks JavaScripti. Selle funktsiooni saab hiljem välja eksportida nupuga *Download Configuration* ja samamoodi käivitada käsurealt, nagu eelnevas peatükis kasutajaliidese konfiguratsiooni failiga. Esmalt on vaja rakendus taas käivitada, seejärel vajutada nupule 3 (joonis 7), et liikuda lehele, kus

on võimalik enda funktsiooni kirjutada. Alguses on väli täidetud näitega, mis formaadis see funktsioon peab olema kirjutatud. Külje peal on ka inglise keeles juhend, kuidas seda kasutada. Esmalt tuleb sisestada taas sensori sõnum *Raw Data* välja. Järgnevalt proovime ära dekodeerida sõnumist temperatuuri ja õhuniiskuse, kuna on teada, mis nende väljade vastused on. Selleks kasutame sisse ehitatud funktsiooni getValue(buffer, a, b), buffer on siis sõnumi väärtus, a on mis baidi pealt soovitud väärtus hakkab ja b on mitu baiti pikk see sõnum on. Temperatuuri väärtuseks on siis getValue(buffer, 4, 2), seekord on vaja vajutada ka nuppu *Decode*, et näha all tulemust, mis peaks olema 2346. Kuna dokumendis on kirjas, et on vaja läbi korrutada 0,01ga, saame muuta rea ümber „getValue(buffer, 4, 2) * 0.01“ ja seejärel uuesti *Decode* vajutada. Samamoodi saab lisada ka õhuniiskuse. Funktsiooni kirjutamine annab võimalusi juurde, nagu tingimuslaused, mida ei ole praeguses faasis võimalik teha kasutajaliidesega. Vaadates joonist 2 on näha, et *device type* on 0x01/0x0B, joonisel 12 näeme, et nendele väärtustele vastab uus väärtus. Seda saame lisada funktsiooni, kasutades ternaarset tehet [18].

DeviceType– 1 byte – Device Type of Device
0x01——R711
0x02——R311A
0x03——RB11E
0x04——R311G

Joonis 12. Seadme väärtused vastavalt sõnumile.

Tulemus peaks olema nagu joonisel 13.

Joonis 13. Kasutaja defineeritud funktsiooni näide.

Kui funktsioon on valmis, siis on võimalik see ka salvestada. Selleks tuleb vajutada nupule *Download Configuration*, see fail taaskord salvestada samasse kausta, kus asub praktilise osa B fail decoderB.js.

Seejärel jooksutada käsurealt käsk „node decoderB.js export1.js AQEBHgkqC2gAAAA=“, mis peaks väljastama sama tulemuse, mis ilmus graafilises kasutajaliideses (vt joonis 14).

```
Jans-MacBook-Pro:praktilineOsaB jandurejko$ node decoderB.js export.js AQEBHgkqC2gAAAA=
{ deviceType: 'R718A', temperature: 23.46, humidity: 29.2 }
```

Joonis 14. Käsurea käsk koos tulemusega.

Siin alapeatükis käsitleti ainult väikest osa sellest, mis oma funktsiooni kirjutamisega teha on võimalik, sest tegemist on JavaScriptiga ja kogu funktsionaalsust ei ole võimalik kirja panna. Edasiarendusena on plaan viia järjest rohkem funktsiooni kirjutamise võimalusi kasutajaliidesesse, kuid mis täpsemalt, sõltub juba Catapult Labsi klientide soovist.

4. Valideerimine

Selles peatükis testitakse mitmete sensorite sõnumeid, kasutades veebirakendust. Veebirakenduse tulemusi võrreldakse seejärel Catapult Labsi testidega, et saada kinnitust andmete õigsuses.
Järgnevas alapeatükis testitakse veebirakendust erinevate sensoritega, kus igal sensoril on kaks sõnumit. Alguses luuakse konfiguratsioonifail kasutajaliidesega, seejärel testitakse teist sõnumit käsurealt, kasutades selleks eelnevalt salvestatud konfiguratsioonifaili.

4.1 Tulemuste ülevaade

Esmaselt testitakse läbi Netvoxi veelekkesensor. Ekraanitõmmis dokumendist on joonisel 15. Kasutades kasutajaliidest, proovitakse saada sõnumist „AQYBHQEBAAAAAAA=“ kätte väljad *Water1Leak* ja *Water2Leak*. Ruumi kokkuhoidmiseks kirjutatakse Catapult Labsi testitulemused tabeli formaadis (vt tabel 4)

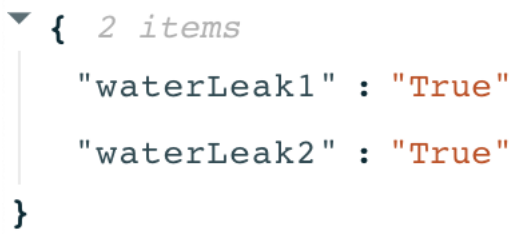
Device	Device Type	ReportType	NetvoxPayLoadData			
R311W	0x06	0x01	Battery(1Byte Unit:0.1V)	Water1Leak (1Byte 0:noleak 1:leak)	Water2Leak(1Byte 0:noleak 1:leak)	Reserved (5Bytes, fixed 0x00)

Joonis 15. Netvoxi veelekke sensori dokument [12].

Tabel 4. Veelekke sensori sõnum koos väärtustega.

Sõnum	**Väärtus**
“AQYBHQEBAAAAAAA=”	waterLeak1 = 1, waterLeak2 = 1
“AQYBHAAAAAAAAAA=”	waterLeak1 = 0, waterLeak2 = 0

Esmalt käivitame veebirakenduse, sisestame sõnumi *Raw Data* välja ja dokumendi järgi seadistame ära read 4 ja 5. *Unit* valime neil *Boolean* – see tähendab, et kui sõnum on 1, siis vastuseks on *True*, ja kui 0, siis vastuseks *False*. Lisades väljadele nimed, peaks ilmuma dekodeeritud sõnum nagu joonisel 16.



Joonis 16. Kasutajaliidese tulemus veelekkesensori sõnumist.

Kui vaatame tabeli 4 esimest rida, siis tulemused ühtivad. Järgnevalt salvestatakse konfiguratsioon ja käivitatakse käsurealt uue sõnumiga, et testida, kas dekodeerimine toimib ka teist korda. Selleks lisatakse fail osa B kausta ja käsurea käsk on „node decoderB.js export.json AQYBHAAAAAAAAAA=“. Tulemus on näha joonisel 17.

```
Jans-MBP:praktilineOsaB jandurejko$ node decoderB.js export.json AQYBHAAAAAAAAAA=
{ name: 'waterLeak1', value: 'False', unit: 'boolean' }
{ name: 'waterLeak2', value: 'False', unit: 'boolean' }
```

Joonis 17. Veelekkesensori dekodeeritud sõnum käsurida kasutades.

Võrreldes seda tabeli 4 tulemusega on näha, et tulemused ühtivad ja veebiliides toimib nagu peab. Eelmise näite veelekkesensor oli samuti Netvox nagu ka temperatuuriandur. Järgnevas näites tehakse läbi teise tootja sensoriga. Selleks kasutame Oyster GPS sensorit. See edastab infot kiiruse ning pikkus- ja laiuskraadide kohta. Sensor edastab ka infot, kas sensor on liikvel, aga kuna hetkel kasutajaliidesega ei saa baite tükeldada bittideks ja nendest infot lugeda, siis selle näite puhul keskendutakse väärtustele, mis tulevad baitidena ehk kiirus ning pikkus- ja laiuskraadid. Sensori sõnum koos tulemustega asub tabelis 5.

Tabel 5. GPS sensori sõnum koos väärtustega.

Sõnum	**Väärtus**
U6t4PAQh+Y6UCrM=	*longitude* = 101.4541139 *latitude* = -189.6275708 *speed* = 10 km/h *battery* = 4.475V
ADDt7AAyJEUAAN4=	*longitude* = -32 *latitude* = 116 *speed* = 0 km/h *battery* = 5.55V

Offset	Description
0 (INT32)	32 bit latitude, **signed**, LSb = 0.000’000’1°. To convert, first calculate the signed integer value, then divide by 10 million to get a floating-point value.
4 (INT32)	32 bit longitude, **signed**, see above
8.0	0: Out of trip, 1: In-trip
8.1	Last fix failed
8.2 - 8.7	Heading, LSb = 5.625°
9 (BYTE)	Speed, LSb = 1 km/h
10 (BYTE)	Battery voltage, LSb = 25 mV

Joonis 18. Oyster GPS sensori dokument.

Kasutades Oysteri dokumenti (vt joonis 18), saame ära täita kasutajaliideses väljad 0, 4, 9 ja 10. Väljadel 0 ja 4 on bait pikkusteks 3 ja *Unit* peab olema „ ° “, sest kraadid saavad olla ainult 180 ja -180 vahel ning kui valida kraadi ühik, siis teisendatakse arvud õigeks. Arvud peab ka 10 000 000 läbi jagama, seega *Multiplier* lahtrisse läheb 0.0000001. *Battery voltage* eeldab, et korrutatakse ka läbi 25ga, seega reale 10 läheb *Multiplier* lahtrisse 25. Kui kõik on õigesti tehtud peaks tulemus ühtima tabeliga 4, nagu ka näha joonisel 19.


```
{ 4 items
  "latitude" : "101.4541139°"
  "longitude" : "-189.6275708°"
  "speed" : "10km/h"
  "volt" : "4475mV"
}
```


Joonis 19. Kasutajaliidese tulemus GPS sensori sõnumist.

Järgnevalt salvestame konfiguratsiooni ja kasutame seda tabeli 4 teise sensori sõnumiga. Selleks kasutame käsku „node decoderB.js export.json ADDt7AAyJEUAAN4=“.

```
Jans-MBP:praktilineOsaB jandurejko$ node decoderB.js export.json ADDt7AAyJEUAAN4=
{ name: 'latitude', value: -32, unit: '°' }
{ name: 'longitude', value: 116, unit: '°' }
{ name: 'speed', value: 0, unit: 'km/h' }
{ name: 'volt', value: 5550, unit: 'mV' }
```

Joonis 20. GPS sensori dekodeeritud sõnum.

Näeme, et tulemused peaaegu ühtivad (vt joonis 20). Ainuke erinevus on väljas *volt*. Põhjuseks on see, et Catapult Labsil on mV teiseldatud ümber V'ks. Aga kuna lähtuti dokumentatsioonist, siis võib Catapult Labsi arvud läbi korrutada 1000ga ja tuleb sama vastus.

Valideerimise käigus testiti läbi kaks sensorit ning mõlemad andsid soovitud tulemused. Aga samas selgus, et kasutajaliidesel on puudujääke, nagu baidi teisaldamine bittideks – hetkel on see võimalik oma funktsiooni kirjutamisega. Lõppeesmärk oleks siiski saada kasutajaliides selliseks, et oma funktsiooni kirjutamist polegi vaja – lisada baidi tükeldamine bitiks ning sealt info lugemine ja lisada tingimuslausete võimalused. Muud funktsionaalsed nõuded said täidetud.

Tabel 6. Funktsiooni jooksmise kiirus

Kordus	**Kulunud aeg funktsiooni jooksutamiseks**
1	0.499 ms
2	0.500 ms
3	0.305 ms
4	0.419 ms
5	0.349 ms
keskmine	0.4324 ms

Rakendust testiti Google Chrome, Mozilla Firefox ja Safari brauseritega ning kõigil tulemused ühtisid ja vigu ei tuvastatud. Lisaks mõõdeti, kui kaua võttis sensori sõnumi dekodeerimine aega. Testimiseks kasutati 16 baidi pikkust sõnumit ja iga tüki pikkus oli 2 baiti ehk kokku 8 sõnumit. Aja võtmiseks lisati koodi sisse Performance[4] meetod, mis mõõtis ajakulu millisekundi täpsusega. Tabelis 6 on välja toodud iga funktsiooni jooksmise peale kulunud aeg ning 5 testi keskmine. Kõige aeglasem on käivitus 2, kus funktsiooni jooksutamiseks kulus 0,5 ms. Tulemuste keskmine oli 0,4324 ms. Kokkuvõtvalt said mittefunktsionaalsed nõuded täidetud.

[4] https://developer.mozilla.org/en-US/docs/Web/API/Performance

4.2 Edasiarenduse võimalused

Hetkeseisuga ei ole rakendus kõige turvalisem, kuna funktsiooni kirjutamisega kaasneb alati turvariske. Võimalik on käivitada XSS (*Cross-site scripting*) [19] otse veebirakenduses ning kuna on võimalus see funktsioon salvestada konfiguratsioonifailiks, siis jooksutatakse neid ka serveris, kus on võimalik teha suuremat kahju, nagu andmebaaside kustutamine. Tegemist on rakendusega, mis on mõeldud majasiseseks kasutamiseks ja mitte kättesaadavaks avalikult ehk see vähendab rünnakute toimumise tõenäosust ning ka mõju. Tulevikus on plaan kindlasti muuta rakendus turvalisemaks ehk eemaldada enda funktsiooni kirjutamine täielikult ning liikuda ainult kasutajaliidese peale. Lisaks turvalisusele on praeguses seisus kasutajaliides võimeline tegelema ainult lihtsate dekodeerimistega ehk sellistega, mis ei vaja baidi tükeldamist bittideks. Hetkel sai loodud rakendus nii, et tulevikus funktsionaalsust juurde lisada on lihtne ja kiire. Sellepärast praeguse töö raames jäeti sisse baasfunktsionaalsus, mis demonstreeriks lahenduse eeliseid ning võimaldaks uurida, kui palju selline veebirakendus tööd lihtsustab. Tulevikus võiks lisada juurde võimalusi vastavalt klientide vajadustele ja nõuetele.

5. Kokkuvõte

Käesoleva bakalaureusetöö eesmärk oli luua veebirakendus, mis võimaldab dekodeerida sensoritest tulevat sõnumit kasutades kasutajaliidest. Samuti anda kasutajale võimalus salvestada sõnumi konfiguratsioon, et hiljem sama sensori sõnumit dekodeerida ilma veebirakenduseta. See tähendab, et (teoreetiliselt) on igat sensorit vaja ainult üks kord konfigureerida ning tulevikus saab valida olemasolevate konfiguratsioonide vahel, kuni on vaja lisada täiesti uus sensor. Eesmärgiks oli luua võimalikult universaalne lahendus, mida oleks mugav kasutada ja saaks kohe näha ka tulemust.

Praktilise osa käigus loodi toimiv veebirakendus, kus on võimalik kasutajaliidesega dekodeerida sensorist tulenevat sõnumit. Praegune versioon toetab lihtsamate (ühe baidi kaupa) dekodeerimise teisenduste tegemist otse veebiliidese kaudu. Keerukamate dekodeerimiste jaoks on kasutajal võimalik ka dekodeerimise funktsioon ise kirjutada, mis võimaldab näiteks ternaarsed tehteid. Lisaks sellele loodi ka käsurea skript, mis võimaldab salvestatud konfiguratsioone kasutada selleks, et sama tüüpi sensoreid automaatselt ning mitteinteraktiivselt dekodeerida.

Töö idee tuli Catapult Labsi soovist sellise rakenduse järele. Edasiarendusena tuleb lisada rakenduse kasutajaliidesele rohkem funktsionaalsust, vastavalt kliendi soovidele ning vajadustele. Lõpuks tuleb see ühendada Catapult Labsi projektidega.

6. Viidatud kirjandus

[1] Evans, D. (2011). The internet of things: How the next evolution of the internet is changing everything. *CISCO white paper*, *1*(2011), 1-11.

[2] Long Range Wireless IoT | 2019 Guide to LoRa and Other LPWAN Technologies. Vaadatud https://www.postscapes.com/long-range-wireless-iot-protocol-lora/ (16.03.19)

[3] Prieto, M. D., Martínez, B., Monton, M., Guillen, I. V., Guillen, X. V., & Moreno, J. A. (2014, July). Balancing power consumption in IoT devices by using variable packet size. In 2014 Eighth International Conference on Complex, Intelligent and Software Intensive Systems (pp. 170-176). IEEE.

[4] McEwen A, Cassimally H. (2014) Designing the Internet of Things. United Kingdom: John Wiley & Sons Ltd..

[5] Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. Future generation computer systems, 29(7), 1645-1660.

[6] Roman, D. H., Conlee, K. D., Abbott, I., Jones, R. P., Noble, A., Rich, N., ... & Costa, D. (2015). The digital revolution comes to US healthcare. New York: Goldman Sachs.

[7] Want, R., Schilit, B. N., & Jenson, S. (2015). Enabling the internet of things. Computer, (1), 28-35.

[8] Mekki, K., Bajic, E., Chaxel, F., & Meyer, F. (2019). A comparative study of LPWAN technologies for large-scale IoT deployment. ICT Express, 5(1), 1-7.

[9] Ray, B. (n.d.). NB-IoT vs. LoRa vs. Sigfox. Vaadatud https://www.link-labs.com/blog/nb-iot-vs-lora-vs-sigfox (12.03.19)

[10] ScadaCore. Online Hex Converter. Vaadatud https://www.scadacore.com/tools/programming-calculators/online-hex-converter/ (18.03.19)

[11] DarkByte. Online converter. Vaadatud https://conv.darkbyte.ru/ (18.03.19)

[12] Netvox. Netvox LoRaWAN Application Command V1.8.2. Suletud dokument

[13] Digital Matter Support. Decoding the Oyster LoRaWAN Payload. Vaadatud https://support.digitalmatter.com/support/solutions/articles/16000069424-decoding-the-oyster-lorawan-payload (21.03.19)

[14] Digital Matter. Digital Matter Q&A. Vaadatud https://www.digitalmatter.com/Why-Digital-Matter/The-DM-Difference (22.03.19)

[15] Facebook. React A JavaScript library for building user interfaces. Vaadatud https://reactjs.org/ (24.03.19)

[16] Feross Aboukhadijeh. The buffer module from node.js, for the browser. Vaadatud https://github.com/feross/buffer (24.03.19)

[17] Joyent. Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Vaadatud https://nodejs.org/en/ (24.03.19)

[18] Mozilla. Conditional (ternary) operaator. Vaadatud https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Conditional_Operator (24.03.19)
[19] PORTSWIGGER. Cross-site scripting. Vaadatud https://portswigger.net/web-security/cross-site-scripting (02.05.19)

Lisad

I Tarkvara

Käesoleva töö tarkvara ei ole saadav avalikult. Soovi korral näha rakendust või selle algkoodi tuleks võtta ühendust autoriga e-maili teel (jan.durejko@tudeng.ut.ee).

II Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, **Jan Durejko**,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose

„IoT seadmete sõnumite dekodeerimise lihtsustamine kasutajaliidesega“

mille juhendaja on, Pelle Jakovits reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi

Jan Durejko

29.04.19